Class 5: Data viz with ggplot

Junlin Ruan (PID: A17839687)

Today we are exploring the **ggplot** package and how to make nice figures in R.

There are lots of ways to make figures and plot in R. These include:

- so called "base" R
- and add on packages like **ggplot2**

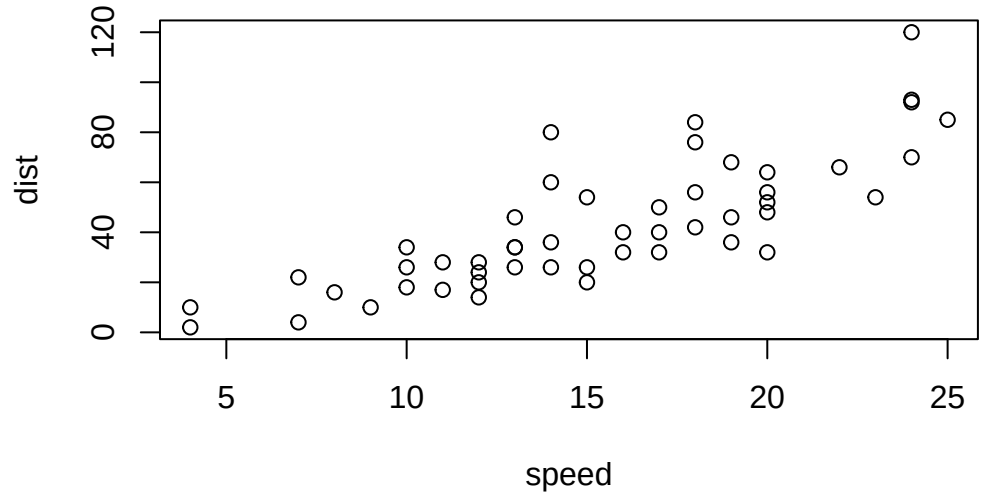
Here is a simple "base" R plot.

```
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

We can simply pass to the 'plot()' function

```
plot(cars)
```



Key-point: Base R is quick but not so nice and simple looking in some folks' eyes.

Let's see how we can plot this with **ggplot2**...

1st I need to install this add-on package. For this we use the `install.packages()` function - **WE DO THIS IN THE CONSOLE, NOT our report**. This is a one time only deal.

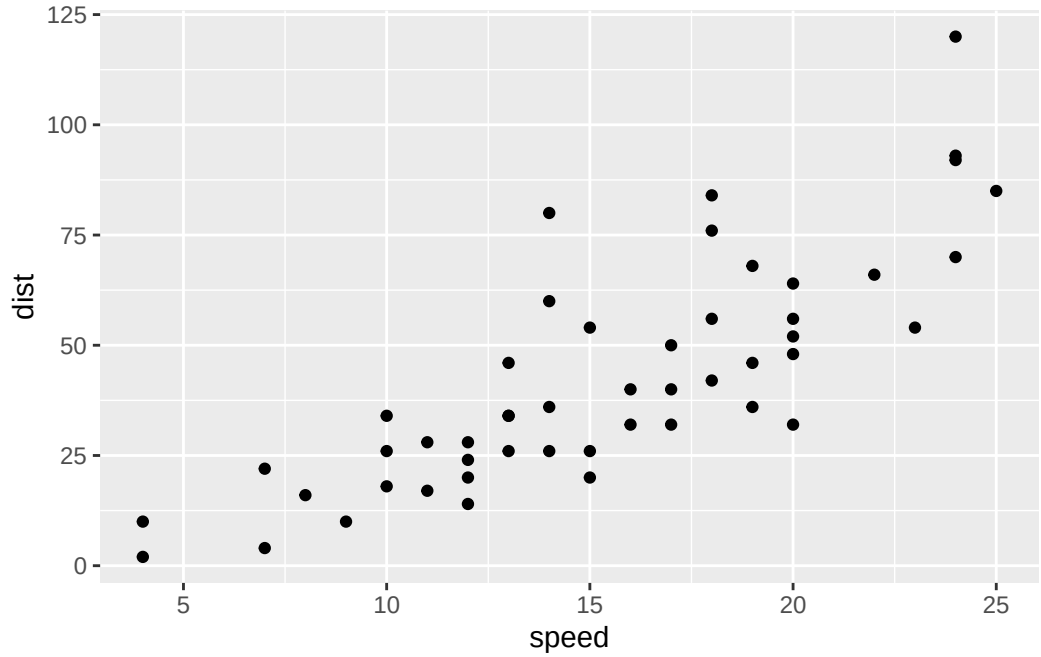
2nd We need to load the package with `library()` function every time we want to use it.

```
library(ggplot2)
ggplot(cars)
```

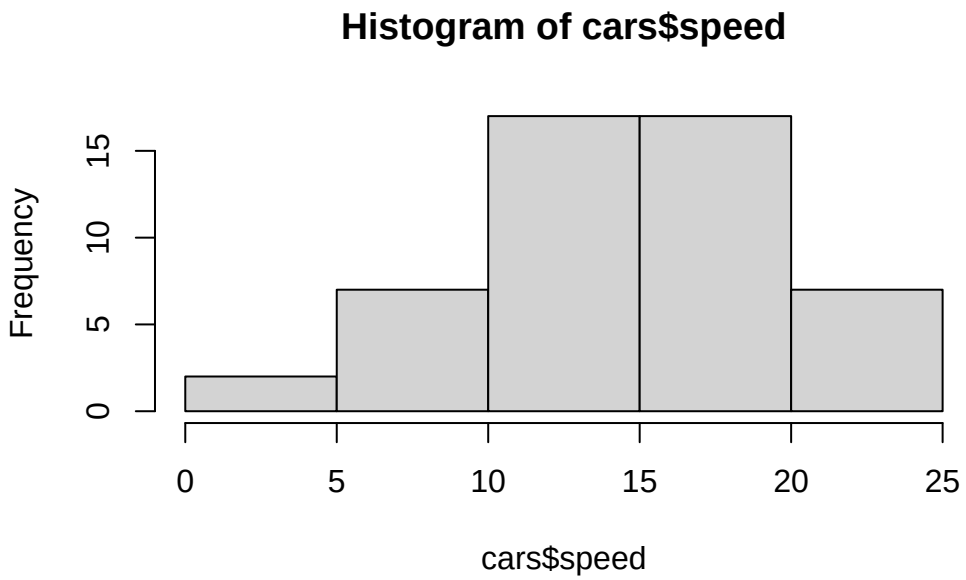
Every ggplot is composed of at least 3 layers:

- **data** (i.e. a data.frame with the things you want to plot),
- aesthetics **aes()** that map the columns of data to your plot features (i.e. aesthitics)
- geoms like **geom_point()** that srt how the plot appears

```
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point()
```




```
hist(cars$speed)
```



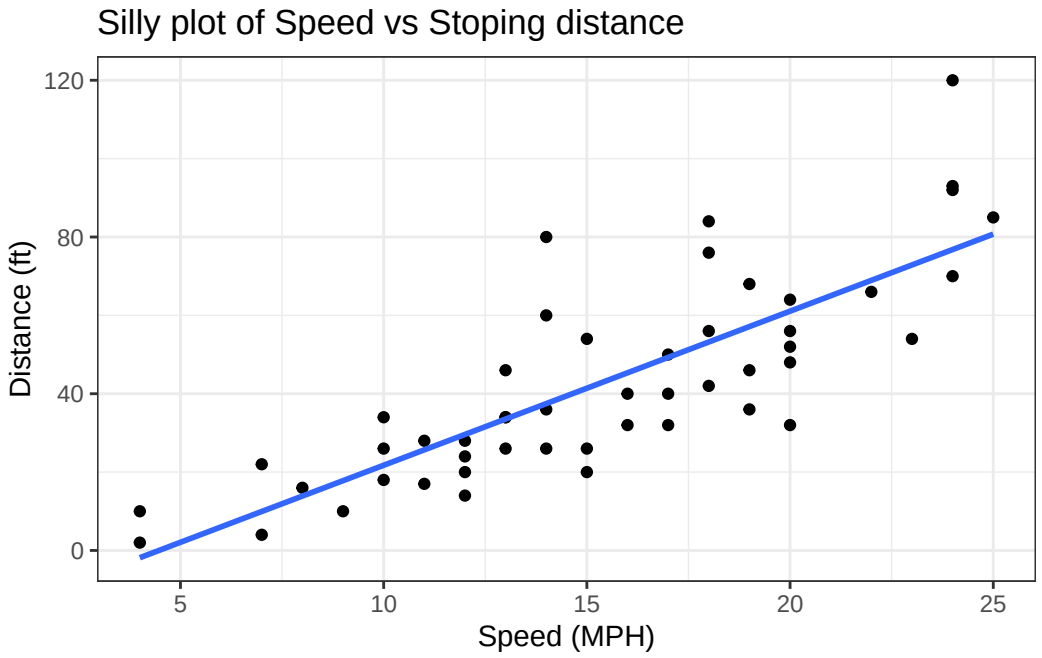
For simple "canned" graphs base R is quicker but as things get more custom and elaborate then ggplot wins out…

Let's add more layers to our ggplot

Add a line showing the relationship between x and y Add a title Add custom axis label "Speed (MPH)" and "Distance (ft)" Change the theme...

```
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)+
  labs(title="Silly plot of Speed vs Stoping distance", x="Speed (MPH)", y="Distance (ft)") +
  theme_bw()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Going further

Read some gene expression data

```
url <- "https://bioboot.github.io/bimm143_S20/class-material/up_down_expression.txt"
genes <- read.delim(url)

head(genes)
```

```
        Gene Condition1 Condition2      State
1      A4GNT -3.6808610 -3.4401355 unchanging
2       AAAS  4.5479580  4.3864126 unchanging
3      AASDH  3.7190695  3.4787276 unchanging
4       AATF  5.0784720  5.0151916 unchanging
5       AATK  0.4711421  0.5598642 unchanging
6 AB015752.4 -3.6808610 -3.5921390 unchanging
```

Q1. How many genes are in this dataset?

```
nrow(genes)
```

```
[1] 5196
```

```
ncol(genes)
```

```
[1] 4
```

Q2. How many "up" regulated genes are there?

```
sum(genes$State == "up")
```

```
[1] 127
```

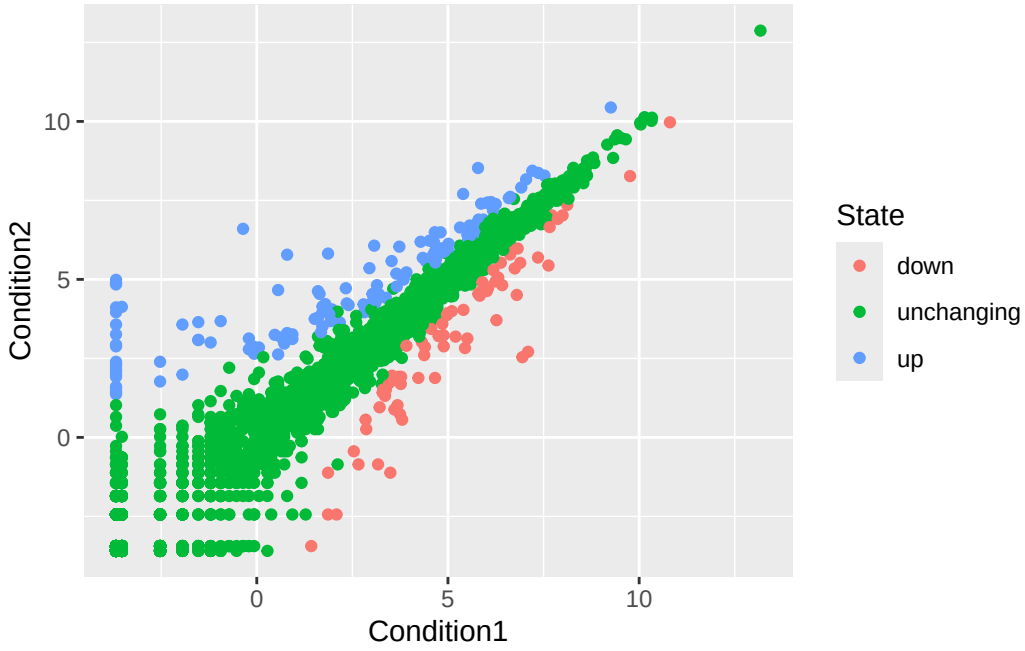
A useful function for counting up occurrences of things in a vector is the `table()` function.

```
table(genes$State)
```

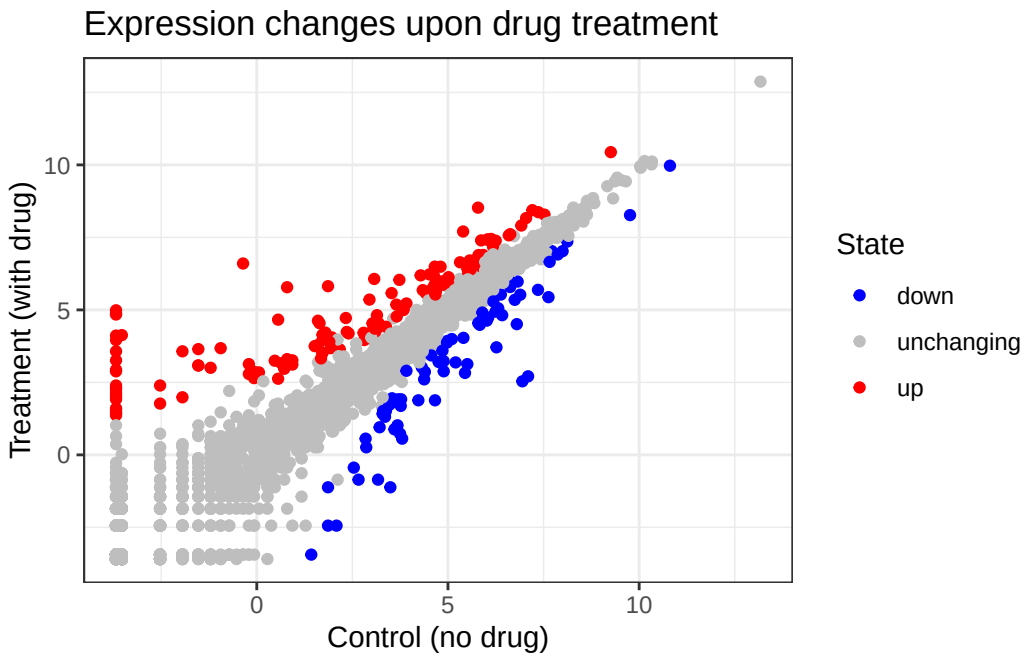
```
      down unchanging         up
        72       4997        127
```

Make a v1 figure

```
p <- ggplot(genes) +
  aes(x=Condition1,
      y=Condition2,
      col=State) +
  geom_point()

p
```




```
p +
  scale_colour_manual(values=c("blue","gray","red"))+
  labs(title="Expression changes upon drug treatment",
       x="Control (no drug)",
       y="Treatment (with drug)") +
  theme_bw()
```



More plotting

Read in the gapminder dataset

```
# File location online
url <- "https://raw.githubusercontent.com/jennybc/gapminder/master/inst/extdata/gapminder.ts
gapminder <- read.delim(url)
```

Lets have a wee peak

```
head(gapminder, 3)
```

```
      country continent year lifeExp      pop gdpPercap
1 Afghanistan      Asia 1952  28.801  8425333  779.4453
2 Afghanistan      Asia 1957  30.332  9240934  820.8530
3 Afghanistan      Asia 1962  31.997 10267083  853.1007
```

Q4. How many different country values are in this dataset?

```
nrow(gapminder)
```

```
[1] 1704
```

```
length(table(gapminder$country))
```

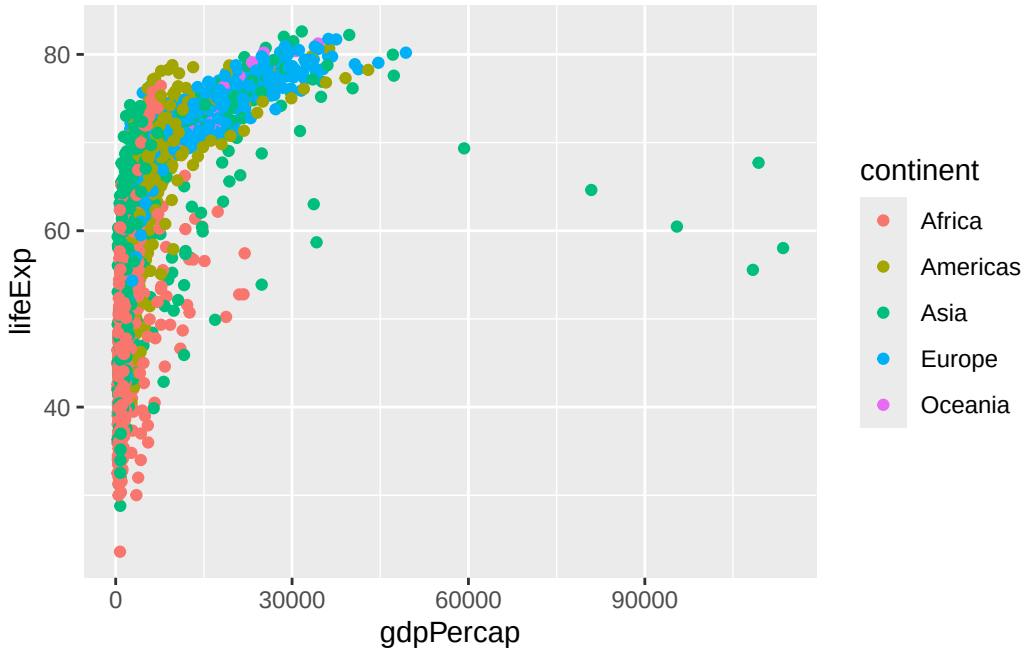
```
[1] 142
```

Q5. How many different continent values are in this dataset?

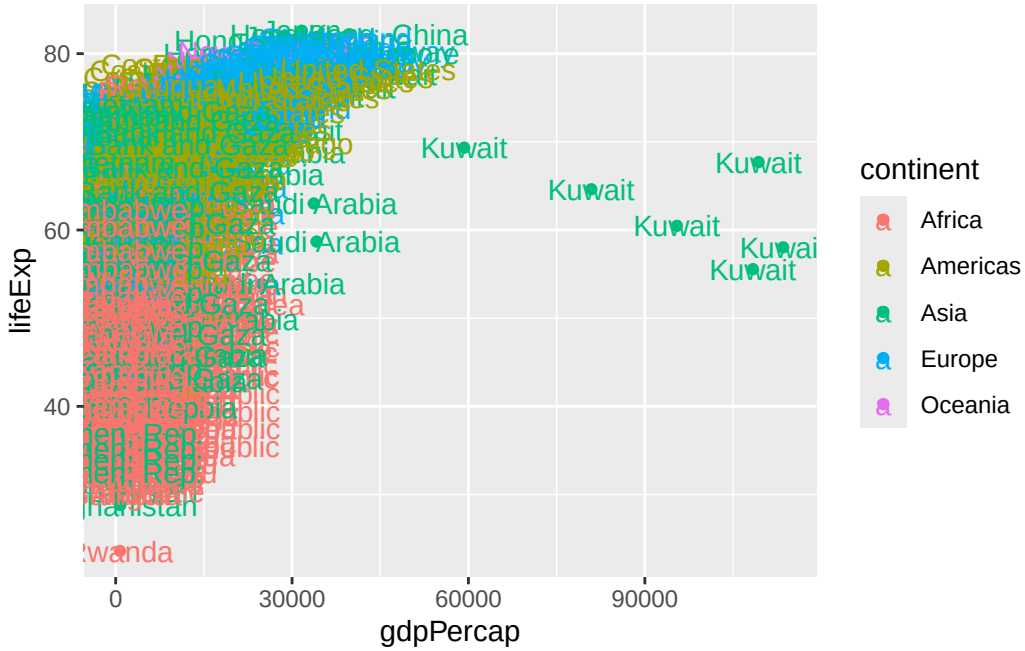
```
unique(gapminder$continent)
```

```
[1] "Asia"     "Europe"   "Africa"   "Americas" "Oceania"
```

```
ggplot(gapminder) +
  aes(gdpPercap, lifeExp, col=continent) +
  geom_point()
```



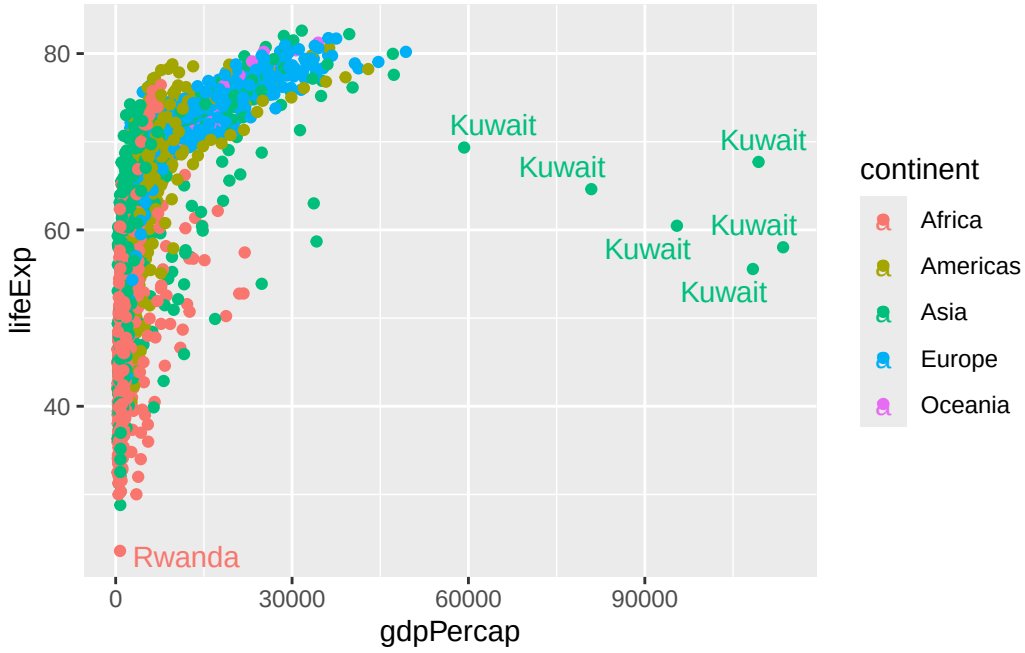
```
ggplot(gapminder) +
  aes(gdpPercap, lifeExp, col=continent, label=country) +
  geom_point() +
  geom_text()
```



I can use the **ggrepel** package to make more sensible labels here.

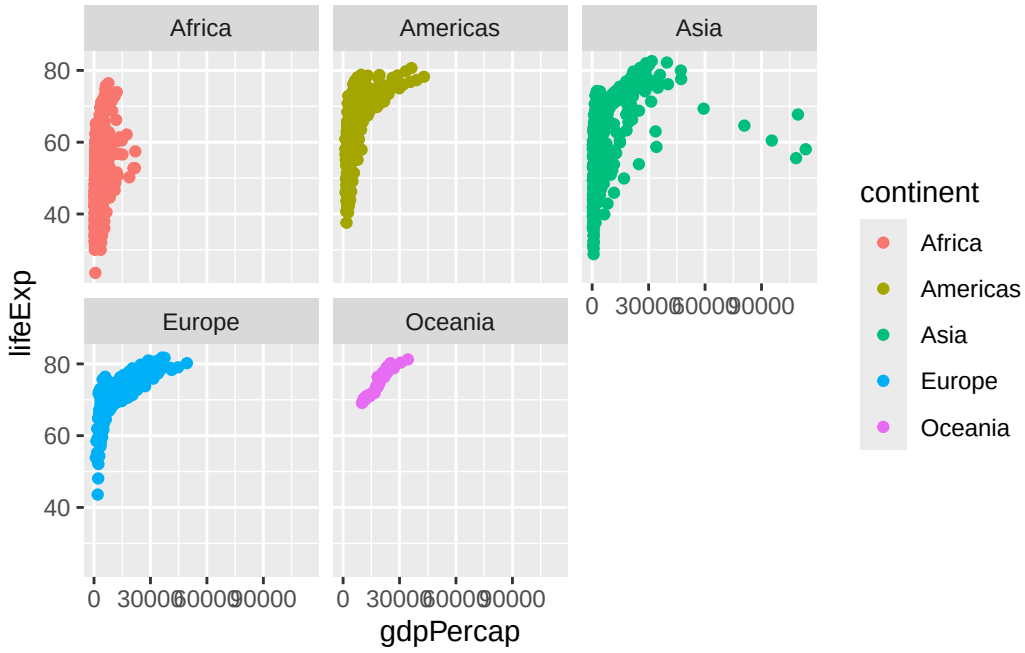
```
library(ggrepel)

ggplot(gapminder) +
  aes(gdpPercap, lifeExp, col=continent, label=country) +
  geom_point() +
  geom_text_repel()
```

```
Warning: ggrepel: 1697 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



I want a separate pannel per continent

```
ggplot(gapminder) +
  aes(gdpPercap, lifeExp, col=continent, label=country) +
  geom_point() +
  facet_wrap(~continent)
```



Summary

The main advantages of ggplot over base R plotting are:

1. **Layered Grammar of Graphics**: ggplot uses a consistent, layered approach where you build plots by adding layers (data, aesthetics, geoms, themes, etc.), making complex plots easier to construct and modify [1], [3], [5].
2. **Sensible Defaults**: ggplot provides attractive default styles and layouts, so your plots look polished without extensive tweaking. Base R requires more manual adjustment for publication-quality visuals [1], [3], [5].
3. **Declarative Syntax**: You specify what you want to show (e.g., which variables map to which aesthetics), rather than how to draw each element. This makes code more readable and easier to maintain [1], [3], [5].
4. **Customization and Scalability**: ggplot makes it straightforward to add layers (like color, shape, smoothing lines, annotations) and to combine multiple plots. Base R can be fiddly and time-consuming for these tasks [1], [3], [5].
5. **Reproducibility**: ggplot code is modular and scriptable, making it easier to reproduce and share visualizations, especially in reports and collaborative projects [1], [3], [5].